A Fast, Preconditioned Conjugate Gradient Toeplitz Solver

Robert Schrieber

March, 1989

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 89.14

NASA Cooperative Agreement Number NCC 2-387

RIACS

Research Institute for Advanced Computer Science

A Fast, Preconditioned Conjugate Gradient Toeplitz Solver

Victor Pan
Robert Schrieber

March, 1989

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 89.14

NASA Cooperative Agreement Number NCC 2-387

A Fast, Preconditioned Conjugate Gradient Toeplitz Solver

Victor Pan*
Computer Science Department, SUNY, Albany, NY 12222
and
Mathematics and Computer Science Department,
Lehman College, CUNY, Bronx, NY 10468

Robert Schreiber†
Research Institute for Advanced Computer Science
Moffett Field, CA 94035 ‡

March 15, 1989

Abstract

We give a simple factorization of an arbitrary hermitian, positive definite matrix in which the factors are well-conditioned, hermitian, and positive definite. In fact, given knowledge of the extreme eigenvalues of the original matrix A, we can achieve an optimal improvement, making the condition numbers of each of the two factors equal to the square root of the condition number of A.

We apply this technique to the solution of hermitian, positive definite Toeplitz systems. Large linear systems with hermitian, positive definite

*Supported by NSF Grant CCR-8805782 and PSC-CUNY Award 668541

†Supported by ONR Contract number N00014-86-K-0610, ARO Grant DAAL03-86-K-0112, and by Cooperative Agreement NCC2-387 between NASA and the Universities Space Research Association.

‡On leave from Rensselaer Polytechnic Institute

Toeplitz matrices arise in some signal processing applications. We give a stable fast algorithm for solving these systems that is based on the preconditioned conjugate gradient method. The algorithm exploits Toeplitz structure to reduce the cost of an iteration to $O(n \log n)$ by applying the fast Fourier Transform to compute matrix-vector products. We use our matrix factorization as a preconditioner.

1 Introduction

We give a simple factorization of an arbitrary hermitian, positive definite matrix A in which the factors are hermitian, positive definite, and are substantially better conditioned than the original matrix A. In fact, given knowledge of the extreme eigenvalues of A, we can achieve an optimal improvement, making the condition numbers of each of the two factors equal to the square root of the condition number of A. The factorization is of the form $A = (A + \mu I)(I - \mu(A + \mu I)^{-1})$. We discuss the optimum choice of μ in Section 3.

Consider the linear system $Ax = b$ where A is an $n \times n$ hermitian, positive definite Toeplitz matrix, $A = [a_{ij}] = [a_{ji}] = [a_{|i-j|}]$. Several direct methods for solving such a system using $O(n^2)$ arithmetic operations are known [18], [14], [12], [3]. In addition, some newer methods that take $O(n \log^2 n)$ operations have been developed [4], [5], [13], [11], [2]. The method of Gragg and Ammar, for example, requires $8n \log^2 n$ real arithmetic operations for a real Toeplitz system. Stability of these methods is discussed by Bunch [6].

Recently, some new attention has been given to the preconditioned conjugate gradient method as a Toeplitz solver. The motivation is that a single iteration of this method can be implemented, using the fast Fourier Transform, at a cost of $O(n \log n)$ arithmetic operations. The hope is that with suitable preconditioners, the number of conjugate gradient iterations can be made small enough to make the method practical. One approach has been to use circulant approximations to A as preconditioners [16], [7]. These work remarkably well for some Toeplitz matrices (finite sections of a singly infinite matrix $A_\infty = [a_{i-j}]$ for which $\sum_{k=0}^{\infty} a_k < \infty$) [8], [17]. As we shall show in Section 4, however, they are not always effective. Here we use instead the general preconditioning strategy discussed above. This strategy is most useful when A is Toeplitz, for in this case, the factor, $(A + \mu I)$, is still

Toeplitz. Conjugate gradient iterations are therefore inexpensive. Moreover its inverse has a representation as the difference between two products of triangular Toeplitz matrices (the Gohberg-Semencul formula) or a triangular Toeplitz matrix and a circulant matrix (the Ammar-Gader formula); this allows iteration with the other factor, $(I - \mu(A + \mu I)^{-1})$ to be carried out cheaply.

In the next section we give a general outline of our method. In Section 3 we give the details of this method, providing an optimum shift parameter μ. We compare it with several competitors in Section 4.

1.1 Notation

For $z \in \mathbf{C}, z^*$ denotes the complex conjugate of z. For a complex matrix A, A^H denotes the conjugate transpose of A, and $\lambda(A)$ denotes the set of eigenvalues of the square matrix A. A matrix A is hermitian, positive definite (hereafter h.p.d.) if $A = A^H$ and for all $\alpha \in \lambda(A), \alpha > 0$. Let A be h.p.d. and let α_1 be the largest and α_n the smallest of A's eigenvalues. Then $\kappa(A) = \alpha_1/\alpha_n$ is called the (spectral) condition number of A.

2 A condition-improving matrix factorization

Let A be a given h.p.d. matrix. Let $\mu \in \mathbf{R}$. Let

$$B \equiv A + \mu I \tag{1}$$

and

$$C \equiv I - \mu B^{-1}. \tag{2}$$

Lemma 1 *Let A be a given h.p.d. matrix. Let B and C be given by (1) and (2). Then $A = BC = CB$. If $-\mu$ is not an eigenvalue of A then both B and C have inverses and $A^{-1} = C^{-1}B^{-1} = B^{-1}C^{-1}$.*

Proof: $A = B - \mu I = B(I - \mu B^{-1}) = (I - \mu B^{-1})B$. Invertibility of B is obvious. And $C = B^{-1}A$ must therefore be invertible too. This proves the lemma.

Let the eigenvalues of A, B, and C be given by

$$\{\alpha_n \leq \cdots \leq \alpha_1\} = \lambda(A)\,,$$

$$\{\beta_n \leq \cdots \leq \beta_1\} = \lambda(B)\,,$$

$$\{\gamma_n \leq \cdots \leq \gamma_1\} = \lambda(C)\,.$$

By (1) and (2) we have

$$\beta_j = \alpha_j + \mu\,, \tag{3}$$

$$\gamma_j = 1 - \mu\beta_j\,. \tag{4}$$

Lemma 2 *Let B and C be given by (1) and (2). Then the condition numbers of B and C are given by*

$$\kappa(B) = \frac{\alpha_1 + \mu}{\alpha_n + \mu} \tag{5}$$

and

$$\kappa(C) = \frac{\alpha_1(\alpha_n + \mu)}{\alpha_n(\alpha_1 + \mu)}\,. \tag{6}$$

Thus, for all $\mu > 0$,

$$\kappa(A) = \kappa(B)\kappa(C)\,. \tag{7}$$

Proof: Immediate.

Lemma 3 *Let $\mu = \sqrt{\alpha_1\alpha_n}$. Then $\kappa(B) = \kappa(C) = \sqrt{\kappa(A)}$.*

Proof: Immediate.

In most cases, this factorization does not lead to a reduction in the cost of inverting A or of solving $Ax = b$. For if we employ an algorithm for inverting B and C, such as Gaussian elimination, that is insensitive to condition number, we might as well have used it on A; if the cost of this algorithm depends on $\log\kappa$, as it does for Newton's method (see, for example, [15]) then we have gained nothing. If the cost exceeds $\log\kappa$, as it does for most iterative methods, we may have gained something. But iteration to solve $Cx = z$ is expensive because we need to solve a system with B at each iteration. Some form of inner/outer iteration method could be used to reduce this cost.

There is a case, however, in which we can solve many systems with B in little more time than it takes to solve one. When A is an h.p.d. Toeplitz matrix, then so is B. Then we may use the Gohberg-Semencul formula, or a recent variant, the Ammar-Gader formula, to represent B^{-1}. This reduces the cost of one iteration of CG for C to $O(n \log n)$.

3 A Fast Toeplitz Solver

Let us apply the matrix factorization of the previous section to the solution of a linear system

$$Ax = b \tag{8}$$

where $A = [a_{ij}]$ is an $n \times n$ h.p.d. Toeplitz matrix, $a_{ij} = a_{ji} = a_{|j-i|}$. We suppose that the extremal eigenvalues of A have been estimated (more on this later on, in Section 5). In this case, the matrix B is also h.p.d. and Toeplitz, is completely defined by its first column, and its inverse can be represented as

$$B^{-1} = L_1 L_1^H - L_2 L_2^H$$

where L_1 and L_2 are lower triangular Toeplitz matrices whose elements depend only on the first column of B^{-1} ([9], [18]). Recently, an alternative representation by Toeplitz-circulant products was given by Ammar and Gader [1]:

$$B^{-1} = L_1 E^T - L_2 E \tag{9}$$

where L_1 and L_2 are again lower triangular Toeplitz, and E is circulant. These factors are again functions only of the first column of B^{-1}.

This leads to the following algorithm:

Algorithm 1:
Input: An h.p.d. Toeplitz matrix A, a vector b, and a shift parameter μ.
Output: $A^{-1}b$.
Method:
(Step 1) Solve $By = e_1$, where $e_1 = (1, 0, \ldots, 0)^H$. Construct L_1, L_2, E satisfying (9).
(Step 2) Solve $Bz = b$.
(Step 3) Solve $Cx = z$. Return x.

We solve the linear systems at Steps 1 and 3 of Algorithm 1 by the conjugate gradient (CG) method [10]. At Step 2, we use the representation (9) constructed at Step 1; there is no iteration and only FFTs of n-vectors are required.

In the CG method, each iteration costs $5n$ flops and one matrix-vector product. For the Toeplitz matrix B in Steps 1 and 2, we compute the matrix vector product Bu by a standard technique of embedding B in a circulant matrix of order $2n$ and appending n zeros to u; a circulant matrix times a vector is a convolution, for which the fast Fourier Transform is used:

$$\begin{aligned} \hat{u} &= F_{2n}([u, 0]); \\ \hat{v} &= \hat{u}. * \hat{B}; \\ v &= W_{2n}(\hat{v}); \\ Bu &= v(1:n); \end{aligned}$$

Here 0 represents the zero vector of order n; F_k and W_k are the forward and inverse discrete Fourier Transform operators on C^k, the operator $.*$ represents elementwise multiplication of two vectors, $v(1:n)$ is the first n components of v, and

$$\hat{B} = F_{2n}([b_0, b_1, \ldots, b_{n-1}, 0, b_{n-1}^*, \ldots, b_1^*]) .$$

Note that $\hat{B}$ is real and this reduces the cost of the elementwise multiplication. The cost of a CG iteration with B is therefore

$$Cost(B) = 4\Phi(n)$$

where $\Phi(n)$ is the cost of an n-point FFT. (This does not include the cost of computing $\hat{B}$, which is computed once and for all.)

For Step 3 above, we use CG, too. We require the product Cu and hence also $B^{-1}u$ at every iteration. Using the Ammar-Gader formula, this cost may be reduced to

$$Cost(C) = 7\Phi(n)$$

(see [1]).

The alternative of using B as a preconditioner in the preconditioned CG method is less attractive than Algorithm 1. We would in effect be solving

$$Cx = B^{-1}b$$

by the conjugate gradient method. Thus, we have an equivalent of Steps 2 and 3 above. But at each iteration, we would have to form Cp for some vector p as $B^{-1}Ap$. The cost of each iteration would therefore be $11\Phi(n)$ rather than $7\Phi(n)$ as it is in our implementation.

3.1 The Optimal Shift

The choice $\mu = \sqrt{\alpha_1 \alpha_n}$ is not optimal. Since each iteration of CG with the matrix C costs roughly 1.75 times as much as a CG step with B, we would be better off to choose μ to make C better conditioned at the expense of worsening the condition number of B. It would be reasonable to choose μ to minimize the estimate of the total work given by

$$(4n_B + 7n_C)\Phi(n),$$

where n_B is the number of CG iterations at Step 1 above, n_C the number of CG iterations at Step 3 above. According to [10], we may model these by

$$n_B = F\sqrt{\kappa(B)} \tag{10}$$

and

$$n_C = F\sqrt{\kappa(C)} \tag{11}$$

with F a constant. Then, by Lemma 2,

$$n_B n_C = F^2\sqrt{\kappa(A)} \equiv M = const.$$

Therefore, for any constant K (and motivated by the estimate above we may think of $K = 7/4$),

$$\begin{aligned} f(n_C) &\equiv n_B + Kn_C \\ &= \frac{M}{n_C} + Kn_C \end{aligned}$$

is minimized at

$$\begin{aligned} n_C &= \sqrt{\frac{M}{K}}, \\ n_B &= \frac{M}{n_C} \\ &= \sqrt{MK} \\ &= Kn_C. \end{aligned} \tag{12}$$

Now we want to choose the best shift parmeter μ. In view of (10), (11), and (12) we choose μ to solve

$$\kappa(B) = K^2\kappa(C)\,, \tag{13}$$

which becomes, using (5) and (6),

$$\alpha_n(\alpha_1^2 + 2\mu\alpha_1 + \mu^2) = K^2\alpha_1(\alpha_n^2 + 2\mu\alpha_n + \mu^2)\,.$$

Now change variables from μ to m, where $\mu = m\sqrt{\alpha_1\alpha_n}$; then m satisfies

$$m^2(K^2\alpha_1 - \alpha_n) + m(2(K^2-1)\sqrt{\alpha_1\alpha_n}) + (K^2\alpha_n - \alpha_1) = 0$$

or, dividing by $\alpha_n > 0$, with $\kappa \equiv \kappa(A) = \alpha_1/\alpha_n$,

$$m^2(K^2\kappa - 1) + m(2(K^2-1)\sqrt{\kappa}) + (K^2 - \kappa) = 0\,.$$

The roots are

$$m_\pm = \frac{-(K^2-1)\sqrt{\kappa} \pm K(\kappa-1)}{K^2\kappa - 1}\,. \tag{14}$$

Note that $m_+ \geq 0$ only when $\kappa \geq K^2$. If $\kappa < K^2$ there is no possibility of satisfying (13) anyway, in view of (7). Thus, we assume that $\kappa \geq K^2$ and that $\mu = m_+\sqrt{\alpha_1\alpha_n} \geq 0$. For $\kappa > K^2$ the root m_+ is monotone increasing with κ and is asymptotic to $1/K$.

Lemma 4 *Let μ be given by $m\sqrt{\alpha_1\alpha_n}$ where $m = m_+$ (see (14) above). Then*

$$\kappa(B) = K\sqrt{\kappa(A)}\,, \tag{15}$$

$$\kappa(C) = K^{-1}\sqrt{\kappa(A)}\,. \tag{16}$$

Proof: According to the derivation of (14), the relation (13) holds. Then, by (7),

$$\begin{aligned}\kappa &= \kappa(B)\kappa(C)\\ &= (\kappa(B)/K)^2\,,\end{aligned}$$

whence (15) follows. And (16) follows from (13) and (15). This proves the lemma.

Now, assume (10) and (11) and choose $\mu = m_+\sqrt{\alpha_1\alpha_n}$ so that (13) holds. In our application, $K = 1.75$. The total cost is then

$$\begin{aligned}(4n_B + 7n_C)\Phi(n) &= 4(n_B + Kn_C)\Phi(n) \\ &= 4(2n_B)\Phi(n) \\ &= 8\sqrt{\kappa(B)}F\Phi(n) \\ &= 8\sqrt{K}\kappa^{1/4}F\Phi(n) \\ &= 4\sqrt{7}\kappa^{1/4}F\Phi(n)\,. \end{aligned} \tag{17}$$

For comparison, let n_{CG} be the number of iterations required by CG for A. We assume the model

$$Cost(CG) = 4n_{CG}\Phi(n) = 4F\Phi(n)\sqrt{\kappa(A)}\,. \tag{18}$$

Compared with (18) we can see that the new method is an improvement for $\kappa \geq 49$.

3.2 Recursive Preconditioning

The factorization $A = BC$ can be used recursively. In particular, let us consider solving the equation $By = e_1$ at Step 1 of Algorithm 1 by using this preconditioned conjugate gradient solver. We now solve for two optimum shift paramemters; μ_1, which we use in solving for y in Step 1, is given by the theory above in which we substitute β_1, β_n, and $\kappa(B)$ for α_1, α_n, and $\kappa(A)$. The other, μ, which we use to define B, is now chosen to minimize the sum of the cost of Step 1, which by (17) is

$$4\sqrt{7}\kappa(B)^{1/4}F\Phi(n)\,,$$

and the cost of Step 3, which is

$$7\kappa(C)^{1/2}F\Phi(n)$$

subject to (7). The solution is to take

$$\kappa(C) = \left(\frac{16\kappa(A)}{49}\right)^{1/3}$$

and

$$\kappa(B) = \left(\frac{7\kappa(A)}{4}\right)^{2/3}.$$

Making the substitution $\mu = m\sqrt{\alpha_1\alpha_n}$ as before leads to a cubic in m that has a positive root for all $\kappa > 49/16$. The overall work becomes

$$9\left(\frac{7}{2}\right)^{2/3} F\Phi(n)\kappa(A)^{1/6}.$$

This is better that $4F\Phi(n)\kappa(A)^{1/2}$ (compare (18)) for $\kappa(A) > 140$ and better than $4\sqrt{7}F\Phi(n)(\kappa(A))^{1/4}$ (compare (17)) for $\kappa(A) > 3222$.

4 Numerical Results and Conclusions

First, we used Algorithm 1 to solve the system (8) where A was as follows. We generated a real time series

$$s_k = 2\sin(x_k) + \sin(4x_k + \pi/6) + \nu_k$$

where the n sample points x_k were uniformly spaced in $[0, 2\pi)$ and ν_k was a Gaussian random variable (white noise) with mean zero and variance (power) σ^2. Next, we took a_j to be the autocorrelation of $\{s_k\}$ with lag j, that is,

$$a_j = \sum_{k=0}^{n-1} s_k s_{k+j}$$

where the index $k + j$ was taken modulo n.

To choose μ, we used the Lanczos algorithm to approximately tridiagonalize A, then computed the extreme eigenvalues τ_1 and τ_m of the resulting $m \times m$ symmetric tridiagonal matrix. We used the approximations

$$\begin{aligned} \alpha_1 &\approx \tau_1, \\ \alpha_n &\approx \tau_m. \end{aligned}$$

Tables 1 and 2 give experimental results for $\sigma = 10$ and $\sigma = 1$. In each table we list the number of sample points, n; the number of CG iterations at Step 1 of Algorithm 1, n_B; the number of CG iterations at Step 3 of

n	n_B	n_C	$n_B + \frac{7}{4}n_C$	n_{CG}	$\|Ax - y\|_2$	$\|Ax_{CG} - y\|$
16	17	7	29	17	1.8(-12)	1.2(-13)
64	39	13	62	46	9.7(-12)	6.9(-12)
256	50	18	82	73	1.2(-11)	8.1(-12)
1024	70	23	110	124	5.3(-11)	2.9(-11)
4096	59	38	126	162	1.2(-10)	6.1(-11)

Table 1: Results for $\sigma = 10$.

n	n_B	n_C	$n_B + \frac{7}{4}n_C$	n_{CG}	$\|Ax - y\|_2$	$\|Ax_{CG} - y\|$
16	17	12	38	18	2.8(-12)	1.9(-13)
64	41	32	97	80	6.6(-12)	7.7(-12)
256	49	47	131	156	2.8(-11)	1.3(-11)
1024	50	75	181	253	8.7(-11)	2.7(-11)
4096	35	217	415	540	1.7(-9)	6.5(-11)

Table 2: Results for $\sigma = 1$.

n	n_B	n_C	$n_B + \frac{7}{4}n_C$	n_{CG}	$\|Ax - y\|_2$	δ
16	18	12	39	18	7.7(-13)	5
64	51	24	93	80	9.0(-12)	5
256	64	34	124	156	2.9(-11)	5
1024	67	54	162	253	7.9(-11)	5
4096	46	169	342	540	1.9(-10)	5
4096	109	73	237	540	4.0(-9)	10

Table 3: Results for $\sigma = 1$. Shift $\mu = \tau_m/\delta$.

Algorithm 1, n_C; the equivalent number of CG iterations taken by Algorithm 1, $n_B + \frac{7}{4}n_C$; the number of iterations taken by unpreconditioned CG, n_{CG}; the residuals achieved by CG and by Algorithm 1. The number of Lanczos iterations that were used to estimate eigenvalues varied from 10 to 40.

In our experience, τ_1 is extremely accurate, but τ_m may be several times larger than α_n. We also found that an underestimate of α_n, which results in a smaller μ than we would take given perfect knowledge and hence a bias in favor of more iterations at Step 1 and fewer at Step 3, is preferable to an overestimate. Thus, we replaced our estimate of α_n by

$$\alpha_n \approx \tau_m/\delta$$

where $\delta > 1$ is a parameter of the algorithm. Results are given in Table 3 for the same class of matrices as above, with $\sigma = 1$. Note that with this better estimate of the optimal shift, we are doing far better than with unpreconditioned CG.

Strang [16] and Chan [7] have advocated circulant preconditioners. Strang takes the circulant $C = [c_{(j-i) \bmod n}]$ with $c_k = a_k, \quad k = 0, \ldots, m$ where $m = n/2$. The other diagonals of C are determined by symmetry and the requirement that C be circulant. Thus C coincides with A in the central half of the diagonals. Chan's choice is to take C to minimize $\|A - C\|_F$, the Frobenius norm of the difference, among all circulant C. For this, one takes $c_k = (ka_{n-k} + (n-k)a_k)/n$. The cost of each iteration is 1.5 times greater than the cost of a CG iteration, since solving the circulant preconditioning system at each step requires both a forward and an inverse FFT of length n.

n	1.5 n_S	1.5 n_C
16	33	30
64	122	114
256	662	293

Table 4: Results for Chan and Strang preconditioners, $\sigma = 1$.

We used these techniques for the model problem above, with $\sigma = 1$. The results, in Table 4, show that neither technique is competitive with ours for this problem. For other problems, in particular when a_k decays with increasing k, we have found them to be superior, as has been predicted by Strang, Chan, and Edelman ([17]).

5 Conclusions

We have demonstrated a preconditioner for Toeplitz systems that achieves a significant speedup when compared with unpreconditioned conjugate gradient method and, for certain problems, is considerably better than other previously proposed preconditioners. The asymptotic complexity is $O(n \log n \kappa^{1/4})$ where κ is the condition number of the problem the exponent $1/4$ may be made smaller at the expense of an increase in the constant factor by applying our technique recursively. Compared with the complexity $(8n \log^2 n)$ of other superfast methods, we can see that for large, well-conditioned problems the present technique may be quite useful.

6 Acknowledgement

We would like to thank Joowan Chun and Tom Kailath for pointing out Reference [1].

References

[1] Gregory Ammar and Paul Gader. A variant of the Gohberg-Semencul formula involving circulant matrices. Submitted to SIAM Journal on Matrix Analysis and Applications.

[2] Gregory S. Ammar and William B. Gragg. Superfast solution of real positive definite Toeplitz systems. SIAM J. Matrix Anal. Appl. 9:1 (1988), pp. 61-76.

[3] E. H. Bareiss. Numerical solution of linear equations with Toeplitz and vector Toeplitz matrices. Numer. Math. 13 (1969), pp. 404-424.

[4] R. R. Bitmead and B. D. O. Anderson. Asymptotically fast solution of Toeplitz and related systems of linear equations. Linear Algebra Appl. 34 (1980), pp. 103-116.

[5] R. P. Brent, F. G. Gustavson and D. Y. Y. Yun. Fast solution of Toeplitz systems of equations and computation of Padé approximants. J. Algorithms 1 (1980), pp. 259-295.

[6] James R. Bunch. Stability of methods for solving Toeplitz systems of equations. SIAM J. Scient. and Stat. Comput. 6:2 (1985), pp. 365-375.

[7] Tony F. Chan. An optimal circulant preconditioner for Toeplitz systems. SIAM J. Scient. and Stat. Comp. 9:4 (1988), pp. 766-771.

[8] Raymond H. Chan and Gilbert Strang. Toeplitz equations by conjugate gradients with circulant preconditioner. SIAM J. Scient. and Stat. Comput. 10:1 (1989), pp. 104-119.

[9] I.C. Gohberg and A.A. Semencul. On the Inversion of Finite Toeplitz Matrices and Their Continuous Analogs. Mat. Issled. (in Russian) 2:1 (1972), pp. 201-233.

[10] G.H. Golub and C.F. van Loan . Matrix Computations. Johns Hopkins Univ. Press, Baltimore, Maryland, 1983

[11] F. de Hoog. A new algorithm for solving Toeplitz systems of equations. Linear Algebra Appl. 88/89 (1987), pp. 122-138.

[12] J. R. Jain. An efficient algorithm for a large Toeplitz set of linear equations. IEEE Trans. Acoust. Speech and Signal Processing 27 (1979), pp. 612-615.